PROYECTO DE PROGRAMACIÓN EN ENSAMBLADOR

ESTRUCTURA DE COMPUTADORES

CURSO 2020/2021

Trujillo Guzmán, Erik

Grupo 2M-B

Índice

1. Histórico de desarrollo del proyecto

En este apartado se describe todo el avance del proyecto. Este apartado está compuesto por subapartados correspondientes a cada una de las versiones del código y para cada una de ellas se detallan las actividades realizadas en cada fecha, con una descripción del trabajo llevado a cabo, así como las posibles dificultades encontradas en cada una de ellas y el tiempo invertido en cada actividad.

El formato de la versión es vx.y, donde x es un número entero que se incrementa cuando se realiza un cambio mayor, e y es un número entero que se incrementa cada vez que se realiza un cambio menor, entendiéndose por cambio menor cualquier cambio realizado a una versión anterior del código, y cambio mayor aquel debido a la corrección del Gestor de Prácticas del proyecto para superar un hito. La versión inicial del código es v1.0.

1.1. v1

v1.0 - 13/06/2020 (4,5 horas):

Definición de macros (0,2 horas): Se han implementado las macros DBNZ, LEA, LOAD, PUSH y POP que se utilizarán a lo largo del proyecto.

Declaración de variables globales (0,3 horas): Se han definido las variables globales nF, IMAGEN, FILTRO, FILTRADA necesarias para el proyecto.

Subrutina Sqrt (2 horas): Se ha implementado la subrutina Sqrt. Para depurar la subrutina y encontrar posibles fallos se recogen los casos de prueba que se detallan en el apartado 2.1. No se han encontrado fallos para las pruebas planteadas.

Subrutina nFiltrados (2 horas): Se ha implementado la subrutina nFiltrados. Para depurar la subrutina y encontrar posibles fallos se recogen los casos de prueba que se detallan en el apartado 2.2.

v1.1 - 14/06/2020 (3,5 horas):

Subrutina Comp (2 horas): Se ha implementado la subrutina Comp parcialmente. No se ha podido completar debido a que no se sabe direccionar a nivel de byte y así poder leer los elementos de cada Imagen.

Subrutina SubMatriz (1,5 horas): Se ha implementado la subrutina SubMatriz parcialmente ya que al igual que con la subrutina Comp no se sabe direccionar a nivel de byte y así poder leer los elementos de cada Imagen.

v1.2 – 19/06/2020 (4,55 horas):

Subrutina ValorPixel (1,5 horas): Se ha implementado la subrutina ValorPixel parcialmente ya que al igual que con la subrutina Comp no se

sabe direccionar a nivel de byte y así poder leer los elementos de cada Imagen.

Subrutina FilPixel (1,75 horas): Se ha implementado la subrutina FilPixel parcialmente debido a que no se sabe cómo gestionar el marco de pila o incluso si es necesario su utilización para este caso.

Definición de subrutinas (0,3 horas): Se ha reservado espacio para las subrutinas restantes Filtro y FiltRec.

Programa Principal (0,5 horas): Se ha implementado un programa principal que hace uso de todas las subrutinas para el caso de las variables globales definidas anteriormente.

Memoria (0,5 horas): Se ha redactado este documento para la entrega del proyecto.

1.2. v2

v2.0 - 05/07/2020 (1 hora):

Subrutina Sqrt (1 hora): Se han añadido pruebas para esta subrutina. Sin embargo, no se obtienen los resultados esperados al llamar a Sqrt con el parámetro 2. Se han corregido los fallos de forma que ahora la subrutina pasa correctamente todas las pruebas.

v2.1 - 22/10/2020 (3,5 horas):

Subrutina Sqrt1d (2 horas): Se ha redefinido la subrutina Sqrt para que cumpla con las normas establecidas en el Enunciado del Proyecto del curso 2020-2021. Se han añadido comentarios tanto al código de la subrutina como al de las pruebas para una mejor comprensión del código y visualización de resultados esperados. También se ha comprobado cómo después de realizar los cambios necesarios se obtienen los resultados esperados al aplicar el juego de ensayo (véase apartado 2.1).

Subrutina Comp (1,5 horas): Se han corregidos fallos referentes a la gestión de paso de parámetros por pila y se han añadido comentarios. Se han añadido pruebas al juego de ensayo (véase apartado 2.3). Sin embargo, no se obtienen los resultados esperados debido a que no se sabe operar a nivel de byte. Se corregirán estos errores en actualizaciones posteriores.

v2.2 - 29/10/2020 (2,85 horas):

Subrutina Sqrt1d (0,25 horas): Se ha corregido la subrutina para que satisfaga el Caso de prueba 1.

Subrutina nFiltrados (0,4 horas): Se ha modificado la subrutina. Se ha añadido el Caso de prueba 4. Ahora se obtienen los resultados esperados al aplicar el juego de ensayo (véase apartado 2.1). Se añaden comentarios al código de casos de prueba de esta subrutina.

Subrutina Comp (0,5 horas): Se ha corregido la subrutina de forma que se consiguen los resultados esperados. Se ha añadido un caso de prueba para esta subrutina.

(0,2 horas): Se añade creación y gestión del marco de pila en las subrutinas Sqrt1d, nFiltrados, Comp y ValorPixel.

Subrutina ValorPixel (1 hora): Se ha terminado de implementar la subrutina. Se ha añadido un caso de prueba para esta subrutina. Sin embargo, no se obtiene los resultados esperados.

Memoria (0,5 horas): Se ha redactado este documento para la entrega del proyecto.

v2.3 - 30/10/2020 (1,8 horas):

Subrutina ValorPixel (1 horas): Se han añadido casos de prueba para esta subrutina. Se ha detectado un fallo en la macro DBNZ que utilizaba el registro r4 para guardar el resultado de la comparación, mientras la subrutina ValorPixel también utilizaba el registro r4 para guardar la dirección de comienzo de la SubImagen. Después de corregirlo, la subrutina funciona correctamente y se obtienen los resultados esperados.

Subrutina SubMatriz (0,5 horas): Se ha avanzado en la implementación de esta subrutina, aunque no está terminada.

Memoria (0,3): Se ha completado este documento para la entrega del primer hito del proyecto.

1.3. v3

v3.0 - 03/11/2020 (1,5 horas):

Subrutina SubMatriz (0,75 horas): Se ha completado la implementación de esta subrutina. Se han añadido casos de prueba para esta subrutina. Sin embargo, no se obtienen los resultados esperados. Se arreglará en la próxima versión.

Subrutina FilPixel (0,5 horas): Se ha avanzado en la implementación de la subrutina FilPixel. Aún está sin completar.

Memoria (0,25 horas): Se han corregido faltas de ortografía. Se ha cambiado el nombre de los subapartados 1.X.

v3.1 - 09/11/2020 (2,75 horas):

Subrutina SubMatriz (1,25 horas): Se ha corregido el código de esta subrutina de forma que se obtienen los resultados esperados para el juego de ensayo (véase apartado 2.4). Se han añadido más casos de prueba para esta subrutina.

Subrutina FilPixel (1,25 horas): Se han añadido casos de prueba para esta subrutina. Se ha terminado de implementar la subrutina FilPixel, aunque no se obtienen los resultados esperados para todos los casos de prueba.

Memoria (0,25 horas): Se ha actualizado este documento con los avances y nuevos casos de prueba para las subrutinas SubMatriz y FilPixel.

v3.2 - 10/11/2020 (1,75 horas):

Se ha añadido el Caso 4 para la subrutina SubMatriz.

Subrutina FilPixel (1,5 horas): Se ha depurado la subrutina FilPixel en busca de errores. Se han corregido varios errores respecto al paso de parámetros en pila a la hora de invocar subrutinas dentro de FilPixel. Aún no se obtienen los resultados esperados. Se ha añadido el Caso 4 al juego de ensayo de esta subrutina (véase apartado 2.6).

Memoria (0,25 horas): Se ha actualizado este documento con los avances y nuevos casos de prueba para las subrutinas SubMatriz y FilPixel.

v3.3 - 12/11/2020 (1,6 horas):

Subrutina FilPixel (0,35 horas): Se ha depurado la subrutina FilPixel y se ha detectado un fallo en la programación del Caso 3, lo que hacía que no se obtuvieran los resultados esperados. Sin embargo, una vez corregido este fallo, se comprueba que la subrutina genera los resultados esperados para el juego de ensayo.

Subrutina Filtro (1 hora): Se ha implementado la subrutina Filtro. Queda pendiente la elaboración de Casos de prueba para esta subrutina y comprobar que funciona como se desea.

Memoria (0,25 horas): Se ha actualizado este documento con los avances y nuevos casos de prueba para la subrutina FilPixel.

v3.4 - 13/11/2020 (2,5 horas):

Subrutina Filtro (1,25 horas): Se ha corregido un fallo que hacía que no se guardara el valor M-1 a lo largo de la ejecución de la subrutina Filtro que hacía que dicha subrutina entrara en un bucle infinito. Se ha añadido el Caso 1 al juego de ensayo de esta subrutina (véase apartado 2.7). Se ha comprobado que se obtienen los resultados esperados al aplicar los casos de prueba a la subrutina Filtro.

Subrutina FiltRec (1 hora): Se ha avanzado en la implementación de la subrutina FiltRec, aunque todavía está incompleta.

Memoria (0,25 horas): Se ha actualizado este documento con los avances y nuevos casos de prueba para la subrutina Filtro, de cara a la entrega del segundo hito.

1.4. v4

v4.0 - 13/12/2020 (3,6 horas):

Subrutina FiltRec (3 horas): Se ha terminado de implementar la subrutina FiltRec. Se han añadido los Casos 1, 2 y 3 al juego de ensayo de esta subrutina (véase apartado 2.8). Al realizar el Caso 1 se comprueba que la subrutina entra en un bucle infinito. Sin embargo, después de corregir errores presentes en el código de la subrutina, se obtienen los resultados esperados para el juego de ensayo.

Variables globales (0,1 horas): Se ha decidido utilizar como única variable global nF, ya que IMAGEN, FILTRO y FILTRADA cambian con cada caso de prueba.

Memoria (0,5 horas): Se ha actualizado este documento con los avances y el juego de ensayo para la subrutina FiltRec, y se han corregido errores ortográficos detectados.

v4.1 - 15/12/2020 (2,25 horas):

Revisión de Código (0,75 horas): Se ha revisado el código para añadir/borrar comentarios para la mejor comprensión del código, se han eliminado líneas de código innecesarias. Además, se ha indentado el código para que resulte más fácil de leer. Finalmente, se ha revisado el apartado Avisos Importantes del Enunciado del Proyecto y se ha comprobado que se respetan las normas que se describen.

Memoria (1,5 horas): Se ha preparado este documento para la entrega final del código, con todos los avances hechos, el juego de ensayo para cada subrutina y las conclusiones una vez terminado el proyecto.

2. Juego de ensayo

En este apartado se recogen todos los casos de prueba que se han utilizado en el proyecto para comprobar el correcto funcionamiento de cada una de las subrutinas que componen el programa.

2.1. Subrutina Sqrt1d

Caso 1. Llamada a Sqrt1d pasándole el parámetro 0. Se obtiene como resultado en r29 = 0 (0x0).

Caso 2. Llamada a Sqrt1d pasándole el parámetro 1. Se obtiene como resultado en r29 = 10 (0xA).

Caso 3. Llamada a Sqrt1d pasándole el parámetro 2. Se obtiene como resultado en r29 = 14 (0xE).

Caso 4. Llamada a Sqrt1d pasándole el parámetro 100, un cuadrado perfecto. Se obtiene como resultado en r29 = 100 (0x64).

Caso 5. Llamada a Sqrt1d pasándole el parámetro 1590, un número mayor que 1000. Se obtiene como resultado en r29 = 398 (0x18E).

Caso 6. Llamada a Sqrt1d pasándole el parámetro 1024, un cuadrado perfecto cercano a 1000. Se obtiene como resultado en r29 = 320 (0x140).

Caso 7. Llamada a Sqrt1d dos veces consecutivas, la segunda pasándole como parámetro el resultado de la primera. La primera llamada recibe como parámetro el número 4. Se obtiene como resultado en r29 = 14 (0xE).

2.2. Subrutina nFiltrados

nF (dirección de memoria 0) = 0xA

Caso 1. Llamada a nFiltrados pasándole el parámetro 0. Se obtiene como resultado que la dirección de memoria 0 contiene un 0.

Caso 2. Llamada a nFiltrados pasándole un entero positivo (0x8). Se obtiene como resultado que la dirección de memoria 0 contiene 0x8.

Caso 3. Llamada a nFiltrados pasándole un entero negativo. Se obtiene como resultado que la dirección de memoria 0 contiene 0xB.

Caso 4. Llama dos veces a nFiltrados. La primera vez pasándole un entero positivo (0x6). La segunda vez pasándole un entero negativo. Se obtiene como resultado que la dirección de memoria 0 contiene 0x7.

2.3. Subrutina Comp

Caso 1. Llamada a Comp pasándole las dos imágenes descritas a continuación, que solo difieren en uno de sus elementos. Se obtiene como resultado en r29 = 330 (0x014A).

```
IMG1:   data 4, 8
        data 0x00000000, 0x00000000
        data 0x00000000, 0x00002100
        data 0x00000000, 0x00000000
        data 0x00000000, 0x00000000

IMG2:   data 4, 8
        data 0x00000000, 0x00000000
        data 0x00000000, 0x00000000
        data 0x00000000, 0x00000000
        data 0x00000000, 0x00000000
```

2.4. Subrutina SubMatriz

Caso 1. Llamada a SubMatriz pasándole la imagen IMG_1 de 3 filas y 3 columnas de la que se debe extraer la subimagen correspondiente al elemento central (i = 1, j = 1). Se obtiene como resultado la subimagen: 10203040 50607080 90FFFFFF.

```
IMG_1:      data 3, 3
            data 0x40302010, 0x80706050, 0x90
```

Caso 2. Llamada a SubMatriz pasándole la imagen IMG_2 de 5 filas y 8 columnas de la que se debe extraer la subimagen correspondiente al pixel en la posición i = 3, j = 6. Se obtiene como resultado la subimagen: 1617181E 1F202627 28FFFFFF.

```
IMG_2:      data 5, 8
            data 0x04030201, 0x08070605
            data 0x0C0B0A09, 0x100F0E0D
            data 0x14131211, 0x18171615
            data 0x1C1B1A19, 0x201F1E1D
            data 0x24232221, 0x28272625
```

Caso 3. Llamada a SubMatriz pasándole la imagen IMG_2 del Caso 2 de la que se debe extraer la subimagen correspondiente al píxel en la posición i = 1, j = 1. Se obtiene como resultado la subimagen: 01020309 0A0B1112 13FFFFFF.

Caso 4. Llamada a SubMatriz pasándole la imagen IMG_3 de 5 filas y 5 columnas de la que se debe extraer la subimagen correspondiente al pixel en la posición i = 2, j = 3. Se obtiene como resultado la subimagen: 03040523 24253334 35FFFFFF.

```
IMG_3:      data 5, 5
            data 0x44332211, 0x03020155
            data 0x22210504, 0x31252423
            data 0x35343332, 0x44434241
            data 0x00000045
```

2.5. Subrutina ValorPixel

Caso 1. Llamada a ValorPixel pasándole una subimagen nula excepto en su elemento central (con valor 0x55) y un filtro identidad. Se obtiene como resultado en r29 = 85 (0x55).

```
SUBIMG1:    data 0x00000000, 0x00000055, 0x00
```

```
MF1:    data 0, 1, 0, 1, 0, 1
        data 0, 1, 1, 1, 0, 1
        data 0, 1, 0, 1, 0, 1
```

Caso 2. Llamada a ValorPixel pasándole una subimagen nula excepto en su elemento central (con valor 0x55) y un filtro que dobla y cambia el signo del elemento no nulo. Se obtiene como resultado en r29 = -170 (0xFFFFFF56).

```
MF2:    data 0, 1, 0, 1, 0, 1
        data 0, 1, 2, 0xFFFFFFFF, 0, 1
        data 0, 1, 0, 1, 0, 1
```

Caso 3. Llamada a ValorPixel pasándole una subimagen no nula y un filtro que devuelve el valor negativo del doble de la suma de los ocho elementos que lo rodean. Se obtiene como resultado en r29 = -320 (0xFFFFFEC0).

```
SUBIMG2:    data 0x13121110, 0x17161514, 0x18
MF3:        data 2, 0xFFFFFFFF, 0xFFFFFFFE, 1, 2, 0xFFFFFFFF
            data 0xFFFFFFFE, 1, 0, 1, 0xFFFFFFFE, 1
            data 2, 0xFFFFFFFF, 0xFFFFFFFE, 1, 2, 0xFFFFFFFF
```

2.6. Subrutina FilPixel

Caso 1. Llamada a FilPixel pasándole la imagen IMG_2 de 5 filas y 8 columnas y un filtro cualquiera para el píxel del borde inferior derecho. Se obtiene como resultado en r29 = 0x28.

```
IMG_2:  data 5, 8
        data 0x04030201, 0x08070605
        data 0x0C0B0A09, 0x100F0E0D
        data 0x14131211, 0x18171615
        data 0x1C1B1A19, 0x201F1E1D
        data 0x24232221, 0x28272625
```

Caso 2. Llamada a FilPixel pasándole la imagen IMG_3 de 5 filas y 5 columnas y un filtro identidad que se aplica al píxel en la posición i = 2, j = 3. Se obtiene como resultado en r29 = 0x24.

```
IMG_3:  data 5, 5
        data 0x44332211, 0x03020155
```

data 0x22210504, 0x31252423

data 0x35343332, 0x44434241

data 0x00000045

MF4: *data 0, 1, 0, 1, 0, 1*

data 0, 1, -5, -5, 0, 1

data 0, 1, 0, 1, 0, 1

Caso 3. Llamada a FilPixel pasándole la imagen IMG_4 de 4 filas y 8 columnas y un filtro que devuelve la media de los ocho elementos que rodean al pixel en la posición i = 2, j = 2. Se obtiene como resultado en r29 = 61 (0x3D).

IMG_4: *data 4, 8*

data 0x44444444, 0x44444444

data 0x33343444, 0x44444444

data 0x44884444, 0x44444444

data 0x44444444, 0x44444444

MF5: *data 1, 8, 1, 8, 1, 8*

data 1, 8, 0, 8, 1, 8

data 1, 8, 1, 8, 1, 8

Caso 4. Llamada a FilPixel pasándole la imagen IMG_4 de 4 filas y 8 columnas del Caso 3 y un filtro que devuelve la suma de los ocho elementos que rodean al pixel en la posición i = 2, j = 2. El resultado se ajusta al valor máximo (255), por lo que se obtiene como resultado en r29 = 255 (0xFF).

MF6: *data 1, 1, 1, 1, 1, 1*

data 1, 1, 0, 1, 1, 1

data 1, 1, 1, 1, 1, 1

2.7. Subrutina Filtro

Caso 1. Llamada a Filtro pasándole la imagen IMG_5 de 4 filas y 8 columnas y el filtro FILTRO0 que multiplica por 4 cada elemento de IMG_5 y le resta tres veces el valor del situado en la misma columna de la fila anterior. Algunos elementos alcanzan el valor máximo y otros el mínimo. Se obtiene como resultado la imagen filtrada: 04000000 08000000 01020304 04050607 11FF4344 44FF4617 21005354 FF005627 31323334 34353637.

```
IMG_5:      data 4, 8
            data 0x04030201, 0x07060504
            data 0x14134211, 0x17168514
            data 0x24232221, 0x27262574
            data 0x34333231, 0x37363534

FILTRADA0:  res 40
            data 0xAAAAAAAA, 0xAAAAAAAA

FILTRO0:    data 0, 1, -3, 1, 0, 1
            data 0, 1, 4, 1, 0, 1
            data 0, 1, 0, 1, 0, 1
```

2.8. Subrutina FiltRec

Caso 1. Llamada a FiltRec pasándole la imagen IMAGEN1 de 5 filas y 5 columnas, FILTRADA1, el filtro FILTRO1, que devuelve para cada píxel la mitad de su valor, NCambios con valor 20 y MaxFiltrados con valor 5. Se obtiene como resultado en r29 = 0 (0x00), la imagen filtrada: 05000000 05000000 0A000A00 0A000000 00000A00 01000A00 00000000 0A000A00 0A000000 y en la dirección 0 hay un 3 (nF = 3).

```
IMAGEN1:    data 5, 5
            data 0x000A000A, 0x0000000A, 0x000A0000
            data 0x000A000A, 0x00000000, 0x000A000A
            data 0x0000000A

FILTRADA1:  res 32
            data 0xFFFFFFFF, 0xFFFFFFFF

FILTRO1:    data 0, 1, 0, 1, 0, 1
            data 0, 1, 1, 2, 0, 1
            data 0, 1, 0, 1, 0, 1
```

Caso 2. Llamada a FiltRec pasándole la imagen IMAGEN2 de 4 filas y 8 columnas, FILTRADA2, el filtro FILTRO2, que sustituye cada píxel por la media de los que están situados en los cuatro vértices de la submatriz que lo rodea, NCambios con valor 0 y MaxFiltrados con valor 2. Se obtiene como resultado en r29 = -1 (0xFFFFFFFF), la imagen filtrada: 04000000 08000000 FF0000FF FF0000FF FF9F7F7F 7F7F9FFF FF9F7F7F 7F7F9FFF FF0000FF y en la dirección 0 hay un 0 (nF = 0).

```
IMAGEN2:    data 4, 8
```

```
                data 0xFF0000FF, 0xFF0000FF, 0xFF0000FF
                data 0xFF0000FF, 0xFF0000FF, 0xFF0000FF
                data 0xFF0000FF, 0xFF0000FF

FILTRADA2:      res 40
                data 0xAAAAAAAA, 0xAAAAAAAA

FILTRO2:        data 1, 4, 0, 4, 1, 4
                data 0, 4, 0, 4, 0, 4
                data 1, 4, 0, 4, 1, 4
```

Caso 3. Llamada a FiltRec pasándole la imagen IMAGEN3 de 4 filas y 4 columnas, FILTRADA3, el filtro FILTRO3, que devuelve para cada píxel la media de los que lo rodean, NCambios con valor 400 y MaxFiltrados con valor 4. Se obtiene como resultado en r29 = 0 (0x00), la imagen filtrada: 04000000 04000000 01020304 1005060D 02121305 20212223 y en la dirección 0 hay un 1 (nF = 1).

```
IMAGEN3:        data 4, 4
                data 0x04030201, 0x0D0E0F10
                data 0x05040302, 0x23222120

FILTRADA3:      res 24
                data 0xAAAAAAAA, 0xAAAAAAAA

FILTRO3:        data 1, 8, 1, 8, 1, 8
                data 1, 8, 0, 8, 1, 8
                data 1, 8, 1, 8, 1, 8
```

3. Conclusiones

Para la realización de este proyecto, compuesto por 8 subrutinas, se han invertido un total de 37,65 horas. El código fuente "filtror21.ens" tiene un tamaño de 1085 LOC (líneas de código) y se han utilizado un total de 27 casos de prueba para comprobar el correcto funcionamiento del código.

Entre las principales dificultades surgidas durante la realización del proyecto se encuentran:

- Operar con bytes al implementar la subrutina Comp. Al principio se pensaba que solo se podía direccionar a nivel de palabra, sin embargo después de mirar el Juego de Instrucciones del MC88110 se comprobó que tanto la instrucción "ld" como "st" tiene la opción ".b" que permite especificar que se quiere trabajar con bytes.
- La gestión de pila y el marco de pila. Al comenzar el proyecto no se entendía la diferencia entre la pila y el marco de pila y en las primeras versiones del código no se gestionaba el marco de pila. Sin embargo, más tarde se comprendió su utilidad y cómo utilizar la pila correctamente para el paso de parámetros y para almacenar datos importantes entre subrutinas.
- Manejo de registros entre subrutinas. Al tener varias subrutinas que utilizan los mismos registros, si una subrutina almacena un dato importante en un registro y luego llama a otra subrutina que utiliza el mismo registro, se obtenían resultados inesperados al terminar la subrutina principal. Por esto, y en relación con el punto anterior, los datos importantes como las direcciones de la imagen de entrada o la imagen filtrada se almacenaron en pila al comprender la cómo manejar la pila.
- Confusión entre instrucción "or" y "add". En ocasiones, el no saber utilizar correctamente estas dos instrucciones ocasionó errores.

La implementación de este proyecto ha hecho que comprenda mejor el funcionamiento del computador y cómo se transforma el código de lenguajes de alto nivel en código máquina que es capaz de entender el computador. También me ha servido para entender el funcionamiento de la pila viendo como cada proceso utiliza la pila para almacenar la dirección de retorno, los argumentos con los que se le ha llamado y las variables locales.

Sin embargo, me ha supuesto un esfuerzo mayor que el empleado en otros proyectos debido a la dificultad del proyecto y a que se utiliza un lenguaje completamente nuevo y difícil de entender al que se tiene poco contacto durante la carrera. Considero que me hubiera más sencillo de programar si se hubieran realizado en clase más ejemplos programando el código en una máquina y ejecutándolo para estudiar los resultados que muestra el emulador. Sobre todo, la implementación de casos de prueba y el manejo del marco de pila.